LECTURE 38

USING BEAUTIFUL SOUP

MCS 275 Spring 2023
Emily Dumas

Not this kind of beautiful soup

LECTURE 38: USING BEAUTIFUL SOUP

Reminders and announcements:

- **Please** complete your course evaluations.

- Project 4 is due 6pm on Friday 28 April

- Remember to install `beautifulsoup4` with

```
python3 -m pip install beautifulsoup4
```

so you'll be ready for Worksheet 15!

HOMEWORK 14

Available now. Due Tuesday at Noon. It's the last homework!

BS4 BASICS

```
soup = bs4.BeautifulSoup(fp_or_str, "html.parser") # parse!
str(soup) # the HTML
soup.prettify() # prettier HTML
soup.title # first (and only) title tag
soup.p  # first p tag
soup.find("p") # first p tag (alternative)
soup.p.em  # first em tag within the first p tag
soup.find_all("a") # list of all a tags
```

WORKING WITH TAGS

```
str(tag) # HTML for this tag and everything inside it
tag.name # name of the tag, e.g. "a" or "ul"
tag.attrs # dict of tag's attributes
tag["href"] # get a single attribute
tag.text # All the text nodes inside tag, concatenated
tag.string # If tag has only text inside it, returns that text
           # But if it has other tags as well, returns None
tag.parent # enclosing tag
tag.contents # list of the children of this tag
tag.children # iterable of children of this tag
tag.banana # first descendant banana tag (sub actual tag name!)
tag.find(...) # first descendant meeting criteria
tag.find_all(...) # descendants meeting criteria
tag.find_next_sibling(...) # next sibling tag meeting criteria
```

SEARCHING

Arguments supported by all the `find*` methods:

```python
tag.find_all(True) # all descendants
tag.find_all("tagname") # descendants by tag name
tag.find_all(href="https://example.com/") # by attribute
tag.find_all(class_="post") # by class
tag.find_all(re.compile("^fig")) # tag name regex match
tag.find_all("a",limit=15) # first 15 a tags
tag.find_all("a",recursive=False) # all a *children*
```

Also work with `find(),find_next_sibling(),...`

SIMULATING CSS

`soup.select(SELECTOR)` returns a list of tags that match a CSS selector, e.g.

```
soup.select(".wide") # all tags of class "wide"

# ul tags within divs of class messagebox
soup.select("div.messagebox ul")
```

There are many CSS selectors and functions we haven't discussed, so this gives a powerful alternative search syntax.

```
# all third elements of unordered lists
soup.select("ul > li:nth-of-type(3)")
```

The CSS selector examples here were based on those in the Beautiful Soup documentation.

SKETCH OF A SCRAPER

```python
from urllib.request import urlopen
from bs4 import BeautifulSoup
import csv

# grab and parse the HTML
with urlopen("https://acme-onions.com/strategy/") as fp:
    soup = BeautifulSoup(fp,"html.parser")

# find the div we care about
plansdiv = soup.find("div",id="secret_plans")

# save all links in that div to a CSV file
with open("plan_links.csv") as outfile:
    writer = csv.writer(outfile)
    writer.writerow(["dest","linktext"])
    for anchor in plansdiv.find_all("a"):
        writer.writerow([anchor["href"], anchor.text])
```

EXAMPLE SCRAPER

Let's try to extract data about the UIC academic calendar.

HTML TABLES

HTML `table` tag represents a table made up of rectangular cells arranged in aligned rows and columns.

Name	Vertices	Edges	Faces
Cube	8	12	6
Octahedron	6	12	8
⋮	⋮	⋮	⋮

HTML TABLES

HTML `table` tag represents a table made up of rectangular cells arranged in aligned rows and columns.

Name	Vertices	Edges	Faces
Cube	8	12	6
Octahedron	6	12	8
:	:	:	:

```
<table>
  ...
</table>
```

HTML TABLES

HTML `table` tag represents a table made up of rectangular cells arranged in aligned rows and columns.

Name	Vertices	Edges	Faces
Cube	8	12	6
Octahedron	6	12	8
⋮	⋮	⋮	⋮

```
<table>
    <tr> ... </tr>
    <tr> ... </tr>
    <tr> ... </tr>
    ...
</table>
```

HTML TABLES

HTML `table` tag represents a table made up of rectangular cells arranged in aligned rows and columns.

Name	Vertices	Edges	Faces
Cube	8	12	6
Octahedron	6	12	8
⋮	⋮	⋮	⋮

```
<table>
    <tr>
        <th> Name </th>
        <th> Vertices </th>
        ...
    </tr>
    <tr>
        <td> Cube </td>
        <td> 8 </td>
    </tr>
    ...
</table>
```

HTML TABLE TAGS

- `table` - entire table
- `tr` - row (inside a table)
- `td` - data cell (inside a row)
- `th` - header cell (inside a row)

SCRAPER TIPS

- Develop using a local snapshot of the HTML

- Avoid complicated transformation at first; try to faithfully extract the data into a structured format

- Be mindful of maintenance cost (e.g. time); keeping a scraper working as a site that changes over time is difficult. Does size/value of data justify it? [1, 2]

- Try to minimize dependence on markup details that seem most likely to change

REFERENCES

- urllib documentation

- The Beautiful Soup documentation is beautifully clear.

REVISION HISTORY

- 2023-04-22 Last year's lecture on this topic finalized
- 2023-04-20 Updated for 2023